```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
pip install agentds-bench matplotlib seaborn scikit-learn
```

```
Collecting agentds-bench
  Downloading agentds_bench-1.3.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: requests>=2.31.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.32.4)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.2.2)
Requirement already satisfied: python-dotenv>=0.15.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (1.1.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (2025
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (20
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agent
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-ben
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-ben
Downloading agentds_bench-1.3.0-py3-none-any.whl (25 kB)
Installing collected packages: agentds-bench
Successfully installed agentds-bench-1.3.0
```

```
# Cell 1: Initiate BenchmarkClient

!pip install lightgbm agentds scikit-learn pandas matplotlib seaborn --quiet

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import lightgbm as lgb
from sklearn.metrics import f1_score, accuracy_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from agentds import BenchmarkClient

# 🔑 Replace with your credentials
client = BenchmarkClient(api_key="adsb_ExIOhZSrLi8gmawYYzZzbfBo_1760800441", team_name="iampratham29-team")

print("✅ Benchmark client initialized successfully.")
```

```
✅ Benchmark client initialized successfully.
```

```
import zipfile
import os

# Define the path to the zip file in Google Drive
zip_file_path = '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip'

# Define the destination path for extraction in the Colab environment
extraction_destination_path = '/content/Healthcare_extracted'

# Create the destination directory if it doesn't exist
if not os.path.exists(extraction_destination_path):
    os.makedirs(extraction_destination_path)

# Check if the file exists and is a zip file
if os.path.exists(zip_file_path):
```

```
    if zipfile.is_zipfile(zip_file_path):
        # Extract the zip file
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(extraction_destination_path)
        print(f"Zip file '{zip_file_path}' extracted to '{extraction_destination_path}' successfully.")
    else:
        print(f"File '{zip_file_path}' is not a valid zip file.")
else:
    print(f"Zip file '{zip_file_path}' not found.")
```

```
Zip file '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip' extracted to '/content/Healthcare_extracted' successful
```

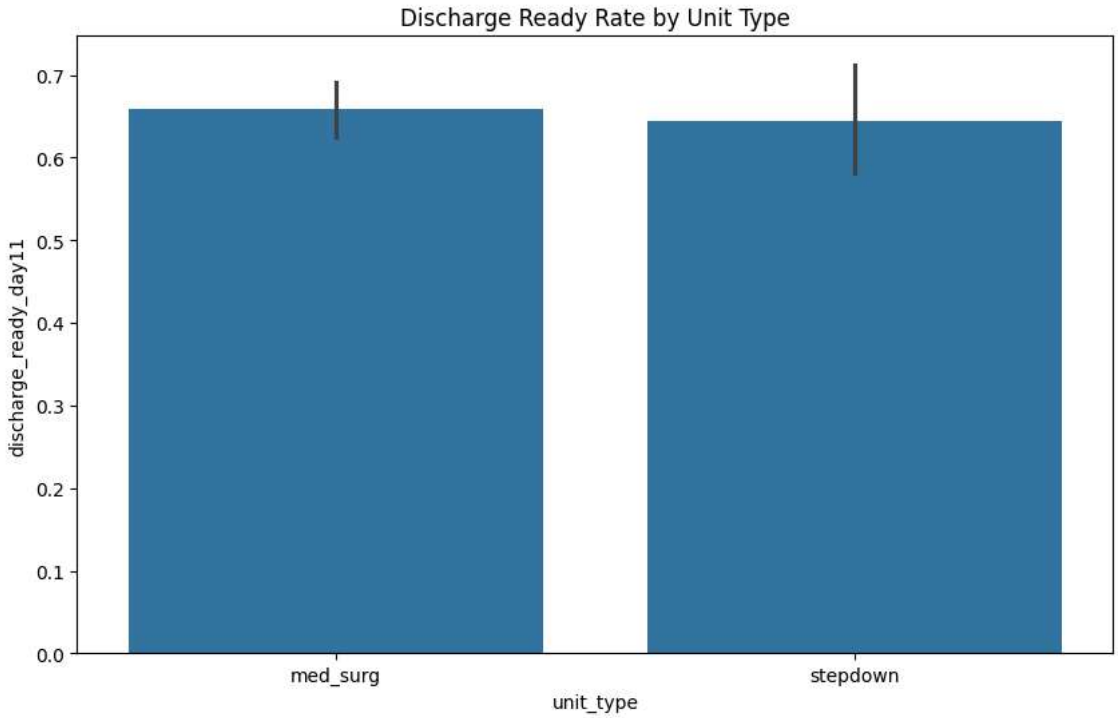
```
# Load hospital stay data
train_stays = pd.read_csv("/content/Healthcare_extracted/stays_train.csv")
test_stays = pd.read_csv("/content/Healthcare_extracted/stays_test.csv")

print(f"Train data shape: {train_stays.shape}")
print(f"Test data shape: {test_stays.shape}")
```

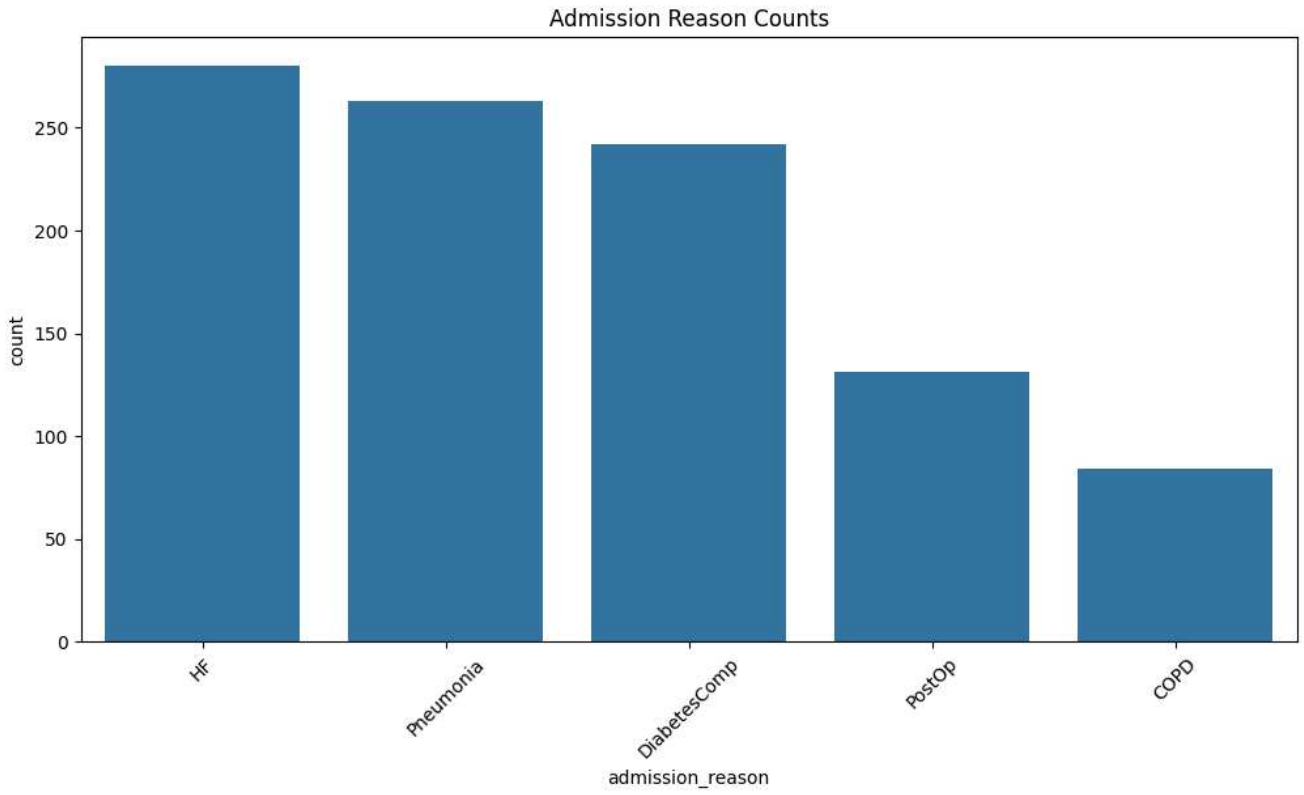
```
Train data shape: (1000, 5)
Test data shape: (1000, 4)
```

```
# Basic augmentation example: add discharge_ready_day11 ratio per unit_type
unit_discharge_rate = train_stays.groupby('unit_type')['discharge_ready_day11'].mean().to_dict()
train_stays['unit_discharge_rate'] = train_stays['unit_type'].map(unit_discharge_rate)
test_stays['unit_discharge_rate'] = test_stays['unit_type'].map(unit_discharge_rate).fillna(train_stays['discharge_ready_day11'
```

```
from sklearn.model_selection import train_test_split

X = train_stays.drop(columns=['discharge_ready_day11'])
y = train_stays['discharge_ready_day11']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

X_val.to_csv("validation_data.csv", index=False)
y_val.to_csv("validation_labels.csv", index=False)
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize discharge readiness distribution by unit_type
plt.figure(figsize=(10,6))
sns.barplot(x='unit_type', y='discharge_ready_day11', data=train_stays)
plt.title('Discharge Ready Rate by Unit Type')
plt.show()

# Admission reason counts
plt.figure(figsize=(12,6))
sns.countplot(x='admission_reason', data=train_stays, order=train_stays['admission_reason'].value_counts().index)
plt.title('Admission Reason Counts')
plt.xticks(rotation=45)
plt.show()
```






```
from sklearn.preprocessing import LabelEncoder

# Clean missing values if any
train_stays.fillna({'unit_type': 'Unknown', 'admission_reason': 'Unknown'}, inplace=True)
test_stays.fillna({'unit_type': 'Unknown', 'admission_reason': 'Unknown'}, inplace=True)

# Label encode categorical features
unit_encoder = LabelEncoder()
reason_encoder = LabelEncoder()

unit_encoder.fit(pd.concat([train_stays['unit_type'], test_stays['unit_type']]))
reason_encoder.fit(pd.concat([train_stays['admission_reason'], test_stays['admission_reason']]))

train_stays['unit_type_encoded'] = unit_encoder.transform(train_stays['unit_type'])
```

```
train_stays['admission_reason_encoded'] = reason_encoder.transform(train_stays['admission_reason'])
test_stays['unit_type_encoded'] = unit_encoder.transform(test_stays['unit_type'])
test_stays['admission_reason_encoded'] = reason_encoder.transform(test_stays['admission_reason'])
```

```
import lightgbm as lgb
from sklearn.metrics import f1_score

features = ['unit_type_encoded', 'admission_reason_encoded', 'unit_discharge_rate']

train_data = lgb.Dataset(train_stays[features], label=train_stays['discharge_ready_day11'])

params = {
    'objective': 'binary',
    'metric': 'binary_logloss',
    'boosting_type': 'gbdt',
    'verbosity': -1,
    'seed': 42
}

model = lgb.train(params, train_data, num_boost_round=200)
```

```
def predict_sample(unit_type, admission_reason):
    unit_encoded = unit_encoder.transform([unit_type])[0]
    reason_encoded = reason_encoder.transform([admission_reason])[0]
    unit_rate = unit_discharge_rate.get(unit_type, train_stays['discharge_ready_day11'].mean())

    sample_df = pd.DataFrame({
        'unit_type_encoded': [unit_encoded],
        'admission_reason_encoded': [reason_encoded],
        'unit_discharge_rate': [unit_rate]
    })

    prediction = model.predict(sample_df)[0]
    print(f"Predicted probability of discharge readiness by day 11: {prediction:.4f}")
    return 1 if prediction >= 0.5 else 0
```

```
# Validation predictions
val_features = X_val.copy()
val_features['unit_type_encoded'] = unit_encoder.transform(val_features['unit_type'])
val_features['admission_reason_encoded'] = reason_encoder.transform(val_features['admission_reason'])
val_features['unit_discharge_rate'] = val_features['unit_type'].map(unit_discharge_rate).fillna(train_stays['discharge_ready_da

val_preds = (model.predict(val_features[features]) >= 0.5).astype(int)

from sklearn.metrics import f1_score

val_f1 = f1_score(y_val, val_preds, average='macro')
print(f"Validation Macro-F1 Score: {val_f1:.4f}")
```

```
Validation Macro-F1 Score: 0.3958
```

```
# Prepare test features
test_stays['unit_discharge_rate'] = test_stays['unit_type'].map(unit_discharge_rate).fillna(train_stays['discharge_ready_day11']

test_features = test_stays[features]
test_preds = (model.predict(test_features) >= 0.5).astype(int)

submission_df = pd.DataFrame({
    'stay_id': test_stays['stay_id'],
    'discharge_ready_day11': test_preds
})

submission_df.to_csv("healthcare_challenge3_predictions.csv", index=False)
print("Predictions saved.")

# Submit predictions
try:
    result = client.submit_prediction("Healthcare", 3, "healthcare_challenge3_predictions.csv")
    if result['success']:
        print("Submission successful!")
        print(f"Score: {result['score']:.4f}")
```

```
        print(f"Metric: {result['metric_name']}")
        print(f"Validation: {'Passed' if result['validation_passed'] else 'Failed'}")
    else:
        print("Submission failed!")
        print(f"Error details: {result.get('details', {}).get('validation_errors', 'Unknown error')}")
except Exception as e:
    print(f"Submission error: {e}. Check your API key and team name.")
```

```
Predictions saved.
✅ Prediction submitted successfully!
📊 Score: 0.3917 (Macro-F1)
✅ Validation passed
Submission successful!
Score: 0.3917
Metric: Macro-F1
Validation: Passed
```

```
#End
```